

One-Pass AUC Optimization

Wei Gao

GAOW@LAMDA.NJU.EDU.CN

*National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China*

Rong Jin

RONGJIN@CSE.MSU.EDU

*Department of Computer Science and Engineering,
Michigan State University, East Lansing, MI, 48824, USA*

Shenghuo Zhu

ZSH@NEC-LABS.COM

NEC Laboratories America, CA, 95014, USA

Zhi-Hua Zhou

ZHOUSH@LAMDA.NJU.EDU.CN

*National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China*

Editor:

Abstract

AUC is an important performance measure and many algorithms have been devoted to AUC optimization, mostly by minimizing a surrogate convex loss on a training data set. In this work, we focus on one-pass AUC optimization that requires going through the training data only once without storing the entire training dataset, where conventional online learning algorithms cannot be applied directly because AUC is measured by a sum of losses defined over pairs of instances from different classes. We develop a regression-based algorithm which only needs to maintain the first and second-order statistics of training data in memory, resulting a storage requirement independent from the size of training data. To efficiently handle high-dimensional data, we develop a randomized algorithm that approximates the covariance matrices by low-rank matrices. We verify, both theoretically and empirically, the effectiveness of the proposed algorithm.

Keywords: AUC optimization, learning to rank, large-scale learning, random projection, square loss

1. Introduction

AUC (Area Under ROC curve) (Metz, 1978; Hanley and McNeil, 1983) is an important performance measure that has been widely used in many tasks (Provost et al., 1998; Cortes and Mohri, 2004; Liu et al., 2009; Flach et al., 2011). Many algorithms have been developed to optimize AUC based on surrogate losses (Herschtal and Raskutti, 2004; Joachims, 2006; Rudin and Schapire, 2009; Kotlowski et al., 2011; Zhao et al., 2011).

In this work, we focus on AUC optimization that requires only one pass of training examples. This is particularly important for applications involving big data or streaming

data in which a large volume of data come in a short time period, making it infeasible to store the entire data set in memory before an optimization procedure is applied. Although many online learning algorithms have been developed to find the optimal solution of some performance measures by only scanning the training data once (Cesa-Bianchi and Lugosi, 2006), few effort addresses one-pass AUC optimization.

Unlike the classical classification and regression problems where the loss function can be calculated on a single training example, AUC is measured by the losses defined over pairs of instances from different classes, making it challenging to develop algorithms for one-pass optimization. An online AUC optimization algorithm was proposed very recently by Zhao et al. (2011). It is based on the idea of reservoir sampling, and achieves a solid regret bound by only storing \sqrt{T} instances, where T is the number of training examples. Ideally, for one-pass approaches, it is crucial that the storage required by the learning process should be independent from the amount of training data, because it is often quite difficult to expect how many data will be received in those applications.

In this work, we propose a regression-based algorithm for one-pass AUC optimization in which a square loss is used to measure the ranking error between two instances from different classes. The main advantage of using the square loss lies in the fact that it only needs to store the first and second-order statistics for the received training examples. Consequently, the storage requirement is reduced to $O(d^2)$, where d is the dimension of data, independent from the number of training examples. To deal with high-dimensional data, we develop a randomized algorithm that approximates the covariance matrix of $d \times d$ by a low-rank matrix. We show, both theoretically and empirically, the effectiveness of our proposal algorithm by comparing to state-of-the-art algorithms for AUC optimization.

Section 2 introduces some preliminaries. Sections 3 proposes the OPAUC (One Pass AUC) framework, and Section 4 provides theoretical analysis and Section 5 presents detailed proofs. Section 6 summaries our experimental results. Section 7 concludes with future work.

2. Preliminaries

We denote by $\mathcal{X} \in \mathbb{R}^d$ an instance space and $\mathcal{Y} = \{+1, -1\}$ the label set, and let \mathcal{D} denote an unknown (underlying) distribution over $\mathcal{X} \times \mathcal{Y}$. A training sample of n_+ positive instances and n_- negative ones

$$\mathcal{S} = \{(\mathbf{x}_1^+, +1), (\mathbf{x}_2^+, +1), \dots, (\mathbf{x}_{n_+}^+, +1), (\mathbf{x}_1^-, -1), (\mathbf{x}_2^-, -1), \dots, (\mathbf{x}_{n_-}^-, -1)\}$$

is drawn identically and independently according to distribution \mathcal{D} , where we do not fix n_+ and n_- before the training sample is chosen. Let $f: \mathcal{X} \rightarrow \mathbb{R}$ be a real-valued function. Then, the AUC of function f on the sample \mathcal{S} is defined as

$$\sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \frac{\mathbb{I}[f(\mathbf{x}_i^+) > f(\mathbf{x}_j^-)] + \frac{1}{2}\mathbb{I}[f(\mathbf{x}_i^+) = f(\mathbf{x}_j^-)]}{n_+ n_-}$$

where $\mathbb{I}[\cdot]$ is the indicator function which returns 1 if the argument is true and 0 otherwise.

Direct optimization of AUC often leads to an NP-hard problem as it can be cast into a combinatorial optimization problem. In practice, it is approximated by a convex optimiza-

tion problem that minimizes the following objective function

$$\mathcal{L}(\mathbf{w}) = \frac{\lambda}{2}|\mathbf{w}|^2 + \sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \frac{\ell(\mathbf{w}^\top(\mathbf{x}_i^+ - \mathbf{x}_j^-))}{2n_+n_-} \quad (1)$$

where ℓ is a convex loss function and λ is the regularization parameter that controls the model complexity. Notice that each loss term $\ell(\mathbf{w}^\top(\mathbf{x}_i^+ - \mathbf{x}_j^-))$ involves two instances from different classes; therefore, it is difficult to extend online learning algorithms for one-pass AUC optimization without storing all the training instances. Zhao et al. (2011) addressed this challenge by exploiting the reservoir sampling technique.

3. The OPAUC Approach

To address the challenge of one-pass AUC optimization, we propose to use the square loss in Eq. (1), that is,

$$\mathcal{L}(\mathbf{w}) = \frac{\lambda}{2}|\mathbf{w}|^2 + \sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \frac{(1 - \mathbf{w}^\top(\mathbf{x}_i^+ - \mathbf{x}_j^-))^2}{2n_+n_-}. \quad (2)$$

The main advantage of using the square loss lies in the fact that it is sufficient to store the first and second-order statistics of training examples for optimization, leading to a memory requirement of $O(d^2)$, which is independent from the number of training examples. Another advantage is that the square loss is consistent with AUC, as will be shown by Theorem 1 (Section 4). In contrast, loss functions such as hinge loss are proven to be inconsistent with AUC (Gao and Zhou, 2012).

As aforementioned, the classical online setting cannot be applied to one-pass AUC optimization because, even if the optimization problem of Eq. (2) has a closed form, it requires going through the training examples multiple times. To address this challenge, we modify the overall loss $\mathcal{L}(\mathbf{w})$ in Eq. (2) (with a little variation) as a sum of losses for individual training instance $\sum_{t=1}^T \mathcal{L}_t(\mathbf{w})$, where

$$\mathcal{L}_t(\mathbf{w}) = \frac{\lambda}{2}|\mathbf{w}|^2 + \frac{\sum_{i=1}^{t-1} \mathbb{I}[y_i \neq y_t](1 - y_t(\mathbf{x}_t - \mathbf{x}_i)^\top \mathbf{w})^2}{2|\{i \in [t-1] : y_i y_t = -1\}|}$$

for sequence $\mathcal{S}_t = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_t, y_t)\}$. It is noteworthy that $\mathcal{L}_t(\mathbf{w})$ is an unbiased estimation to $\mathcal{L}(\mathbf{w})$ for i.i.d. sequence \mathcal{S}_t . For notational simplicity, we denote by X_t^+ and X_t^- the sets of positive and negative instances in the sequence \mathcal{S}_t , respectively, and we further denote by T_t^+ and T_t^- their respective cardinalities. Also, we set $\mathcal{L}_t(\mathbf{w}) = 0$ for $T_t^+ T_t^- = 0$.

If $y_t = 1$, we calculate the gradient as

$$\nabla \mathcal{L}_t(\mathbf{w}) = \lambda \mathbf{w} + \mathbf{x}_t \mathbf{x}_t^\top \mathbf{w} - \mathbf{x}_t + \sum_{i: y_i = -1} \frac{\mathbf{x}_i + (\mathbf{x}_i \mathbf{x}_i^\top - \mathbf{x}_i \mathbf{x}_t^\top - \mathbf{x}_t \mathbf{x}_i^\top) \mathbf{w}}{T_t^-}. \quad (3)$$

It is easy to observe that

$$c_t^- = \sum_{i: y_i = -1} \frac{\mathbf{x}_i}{T_t^-} \quad \text{and} \quad S_t^- = \sum_{i: y_i = -1} \frac{\mathbf{x}_i \mathbf{x}_i^\top - \mathbf{c}_t^- [\mathbf{c}_t^-]^\top}{T_t^-}$$

Algorithm 1 The OPAUC Algorithm**Input:** The regularization parameter $\lambda > 0$ and stepsizes $\{\eta_t\}_{t=1}^T$.**Initialization:** Set $T_0^+ = T_0^- = 0$, $\mathbf{c}_0^+ = \mathbf{c}_0^- = \mathbf{0}$, $\mathbf{w}_0 = \mathbf{0}$ and $\Gamma_0^+ = \Gamma_0^- = [\mathbf{0}]_{d \times u}$ for some $u > 0$

```

1: for  $t = 1, 2, \dots, T$  do
2:   Receive a training example  $(\mathbf{x}_t, y_t)$ 
3:   if  $y_t = +1$  then
4:      $T_t^+ = T_{t-1}^+ + 1$  and  $T_t^- = T_{t-1}^-$ ;
5:      $\mathbf{c}_t^+ = \mathbf{c}_{t-1}^+ + \frac{1}{T_t^+}(\mathbf{x}_t - \mathbf{c}_{t-1}^+)$  and  $\mathbf{c}_t^- = \mathbf{c}_{t-1}^-$ ;
6:     Update  $\Gamma_t^+$  and  $\Gamma_t^- = \Gamma_{t-1}^-$ ;
7:     Calculate the gradient  $\hat{g}_t(\mathbf{w}_{t-1})$ 
8:   else
9:      $T_t^- = T_{t-1}^- + 1$  and  $T_t^+ = T_{t-1}^+$ ;
10:     $\mathbf{c}_t^- = \mathbf{c}_{t-1}^- + \frac{1}{T_t^-}(\mathbf{x}_t - \mathbf{c}_{t-1}^-)$  and  $\mathbf{c}_t^+ = \mathbf{c}_{t-1}^+$ ;
11:    Update  $\Gamma_t^-$  and  $\Gamma_t^+ = \Gamma_{t-1}^+$ ;
12:    Calculate the gradient  $\hat{g}_t(\mathbf{w}_{t-1})$ 
13:   end if
14:    $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \hat{g}_t(\mathbf{w}_{t-1})$ 
15: end for

```

correspond to the mean and covariance matrix of negative class, respectively; thus, Eq. (3) can be further simplified as

$$\nabla \mathcal{L}_t(\mathbf{w}) = \lambda \mathbf{w} - \mathbf{x}_t + \mathbf{c}_t^- + (\mathbf{x}_t - \mathbf{c}_t^-)(\mathbf{x}_t - \mathbf{c}_t^-)^\top \mathbf{w} + S_t^- \mathbf{w}. \quad (4)$$

In a similar manner, we calculate the following gradient for $y_t = -1$:

$$\nabla \mathcal{L}_t(\mathbf{w}) = \lambda \mathbf{w} + \mathbf{x}_t - \mathbf{c}_t^+ + (\mathbf{x}_t - \mathbf{c}_t^+)(\mathbf{x}_t - \mathbf{c}_t^+)^\top \mathbf{w} + S_t^+ \mathbf{w} \quad (5)$$

where

$$\mathbf{c}_t^+ = \sum_{i: y_i=1} \frac{\mathbf{x}_i}{T_t^+} \quad \text{and} \quad S_t^+ = \sum_{i: y_i=1} \frac{\mathbf{x}_i \mathbf{x}_i^\top - \mathbf{c}_t^+ [\mathbf{c}_t^+]^\top}{T_t^+}$$

are the covariance matrix and mean of positive class, respectively.

The storage cost for keeping the class means (\mathbf{c}_t^+ and \mathbf{c}_t^-) and covariance matrices (S_{t-1}^+ and S_{t-1}^-) is $O(d^2)$. Once we get the gradient $\nabla \mathcal{L}_t(\mathbf{w})$, by theory of stochastic gradient descent, the solution can be updated by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \mathcal{L}_t(\mathbf{w}_t)$$

where η_t is the stepsize for the t -th iteration.

Algorithm 1 highlights the key steps of the proposed algorithm. We initialize $\Gamma_0^- = \Gamma_0^+ = [\mathbf{0}]_{d \times d}$, where $u = d$. At each iteration, we set $\Gamma_t^+ = S_t^+$ and $\Gamma_t^- = S_t^-$, and update Γ_t^+ (Line 6) and Γ_t^- (Line 11), respectively, by using the following equations

$$\begin{aligned} \Gamma_t^+ &= \Gamma_{t-1}^+ + \frac{\mathbf{x}_t \mathbf{x}_t^\top - \Gamma_{t-1}^+}{T_t^+} + \mathbf{c}_{t-1}^+ [\mathbf{c}_{t-1}^+]^\top - \mathbf{c}_t^+ [\mathbf{c}_t^+]^\top, \\ \Gamma_t^- &= \Gamma_{t-1}^- + \frac{\mathbf{x}_t \mathbf{x}_t^\top - \Gamma_{t-1}^-}{T_t^-} + \mathbf{c}_{t-1}^- [\mathbf{c}_{t-1}^-]^\top - \mathbf{c}_t^- [\mathbf{c}_t^-]^\top. \end{aligned}$$

Finally, the stochastic gradient $\hat{g}_t(\mathbf{w}_{t-1})$ of Lines 7 and 12 in Algorithm 1 are given by $\nabla \mathcal{L}_t(\mathbf{w}_{t-1})$ that are calculated by Eqs. (4) and (5), respectively.

Dealing with High-Dimensional Data. One limitation of the approach in Algorithm 1 is that the storage cost of the two covariance matrices S_t^+ and S_t^- is $O(d^2)$, making it unsuitable for high-dimensional data. We tackle this by developing a randomized algorithm that approximates the covariance matrices by low-rank matrices. We are motivated by the observation that S_t^+ and S_t^- can be written, respectively, as

$$\begin{aligned} S_t^+ &= \frac{1}{T_t^+} \left(X_t^+ - \mathbf{c}_t^+ \mathbf{1}_{T_t^+}^\top \right) I_{T_t^+} \left(X_t^+ - \mathbf{c}_t^+ \mathbf{1}_{T_t^+}^\top \right)^\top, \\ S_t^- &= \frac{1}{T_t^-} \left(X_t^- - \mathbf{c}_t^- \mathbf{1}_{T_t^-}^\top \right) I_{T_t^-} \left(X_t^- - \mathbf{c}_t^- \mathbf{1}_{T_t^-}^\top \right)^\top, \end{aligned}$$

where I_t is an identity matrix of size $t \times t$ and $\mathbf{1}_t$ is an all-one vector of size t . To approximate S_t^+ and S_t^- , we approximate the identity matrix I_t by a matrix of rank $\tau \ll d$. To this end, we randomly sample $\mathbf{r}_i \in \mathbb{R}^\tau, i = 1, \dots, t$ from a Gaussian distribution $\mathcal{N}(0, I_\tau)$, and approximate I_t by $R_t R_t^\top$, where $R_t = \frac{1}{\tau} (\mathbf{r}_1, \dots, \mathbf{r}_t)^\top \in \mathbb{R}^{t \times \tau}$. We further divide R_t into two matrices where $R_t^+ \in \mathbb{R}^{T_t^+ \times \tau}$ and $R_t^- \in \mathbb{R}^{T_t^- \times \tau}$ that contain the subset of the rows in R_t corresponding to all the positive and negative instances received before the t -th iteration, respectively. Therefore, the covariance matrices S_t^+ and S_t^- can be approximated, respectively, by

$$\hat{S}_t^+ = \frac{1}{T_t^+} Z_t^+ [Z_t^+]^\top - \hat{\mathbf{c}}_{t-1}^+ [\hat{\mathbf{c}}_{t-1}^+]^\top \quad \text{and} \quad \hat{S}_t^- = \frac{1}{T_t^-} Z_t^- [Z_t^-]^\top - \hat{\mathbf{c}}_{t-1}^- [\hat{\mathbf{c}}_{t-1}^-]^\top$$

where

$$\begin{aligned} Z_t^+ &= X_t^+ R_t^+, \quad \hat{\mathbf{c}}_t^+ = \mathbf{c}_t^+ \mathbf{1}_{T_t^+}^\top R_t^+ / T_t^+ \\ Z_t^- &= X_t^- R_t^-, \quad \hat{\mathbf{c}}_t^- = \mathbf{c}_t^- \mathbf{1}_{T_t^-}^\top R_t^- / T_t^-. \end{aligned}$$

Based on approximate covariance matrix \hat{S}_t^\pm , the approximation algorithm essentially tries to minimize $\sum_{t=1}^T \hat{\mathcal{L}}_t(\mathbf{w})$, where

$$\hat{\mathcal{L}}_t(\mathbf{w}) = \mathbf{w}^\top (\mathbf{c}_{t-1}^- - \mathbf{x}_t) + \frac{1}{2} (1 + \mathbf{w}^\top \hat{S}_t^- \mathbf{w}) + \frac{\lambda}{2} |\mathbf{w}|^2 + \frac{1}{2} \mathbf{w}^\top (\mathbf{x}_t - \mathbf{c}_{t-1}^-) (\mathbf{x}_t - \mathbf{c}_{t-1}^-)^\top \mathbf{w} \quad (6)$$

if $y_t = 1$; otherwise,

$$\hat{\mathcal{L}}_t(\mathbf{w}) = \mathbf{w}^\top (\mathbf{x}_t - \mathbf{c}_{t-1}^+) + \frac{1}{2} (1 + \mathbf{w}^\top \hat{S}_t^+ \mathbf{w}) + \frac{\lambda}{2} |\mathbf{w}|^2 + \frac{1}{2} \mathbf{w}^\top (\mathbf{x}_t - \mathbf{c}_{t-1}^+) (\mathbf{x}_t - \mathbf{c}_{t-1}^+)^\top \mathbf{w}. \quad (7)$$

Further, we have the following recursive formulas:

$$Z_t^+ = Z_{t-1}^+ + \mathbf{x}_t \mathbf{r}_t^\top \mathbb{I}[y_t = +1] / \sqrt{m}, \quad (8)$$

$$Z_t^- = Z_{t-1}^- + \mathbf{x}_t \mathbf{r}_t^\top \mathbb{I}[y_t = -1] / \sqrt{m}. \quad (9)$$

It is important to notice that we do not need to calculate and store the approximate covariance matrices \hat{S}_t^+ and \hat{S}_t^- explicitly. Instead, we only need to maintain matrices Z_t^+ and

Z_t^- in memory. This is because the stochastic gradient $\hat{g}_t(\mathbf{w})$ based on the approximate covariance matrices can be computed directly from Z_t^+ and Z_t^- . More specifically, $\hat{g}_t(\mathbf{w})$ is computed as

$$\hat{g}_t(\mathbf{w}) = \mathbf{c}_{t-1}^- - \mathbf{x}_t + \lambda \mathbf{w} + (\mathbf{x}_t - \mathbf{c}_{t-1}^-)(\mathbf{x}_t - \mathbf{c}_{t-1}^-)^\top \mathbf{w} + \left(Z_t^- [Z_t^-]^\top / T_t^- - \hat{\mathbf{c}}_{t-1}^- [\hat{\mathbf{c}}_{t-1}^-]^\top \right) \mathbf{w} \quad (10)$$

for $y_t = 1$; otherwise

$$\hat{g}_t(\mathbf{w}) = \mathbf{x}_t - \mathbf{c}_{t-1}^+ + \lambda \mathbf{w} + (\mathbf{x}_t - \mathbf{c}_{t-1}^+)(\mathbf{x}_t - \mathbf{c}_{t-1}^+)^\top \mathbf{w} + \left(Z_t^+ [Z_t^+]^\top / T_t^+ - \hat{\mathbf{c}}_{t-1}^+ [\hat{\mathbf{c}}_{t-1}^+]^\top \right) \mathbf{w}. \quad (11)$$

We require a memory of $O(\tau d)$ instead of $O(d^2)$ to calculate $\hat{g}_t(\mathbf{w})$ by using the trick $A[A]^\top \mathbf{w} = A([A]^\top \mathbf{w})$, where $A \in \mathbb{R}^{d \times 1}$ or $\mathbb{R}^{d \times \tau}$.

To implement the approximate approach, we initialize $\Gamma_0^- = \Gamma_0^+ = [\mathbf{0}]_{d \times \tau}$ in Algorithm 1, where $u = \tau$. At each iteration, we set $\Gamma_t^+ = Z_t^+$ and $\Gamma_t^- = Z_t^-$, and compute the gradient $\hat{g}_t(\mathbf{w}_{t-1})$ of Lines 7 and 12 in Algorithm 1 by Eqs. (10) and (11), respectively. Γ_t^+ and Γ_t^- are updated by Eqs. (8) and (9), respectively.

Remark. An alternative approach for the high-dimensional case is through the random projection (Johnstone, 2006; Hsu et al., 2012). Let $H \in \mathbb{R}^{d \times \tau}$ be a random Gaussian matrix, where $\tau \ll d$. By performing random projection using H , we compute a low-dimensional representation for each instance \mathbf{x}_t as $\hat{\mathbf{x}}_t = H^\top \mathbf{x}_t \in \mathbb{R}^\tau$ and will only maintain covariance matrices of size $\tau \times \tau$ in memory. Despite that it is computationally attractive, this approach performs significantly worse than the randomized low-rank approximation algorithm, according to our empirical study. This may owe to the fact that the random projection approach is equivalent to approximating $S_t^\pm = I_d S_t^\pm I_d$ by $HH^\top S_t^\pm HH^\top$, which replaces both the left and right identity matrices of S_t^\pm with HH^\top . In contrast, our proposed approach only approximates one identity matrix in S_t^\pm , making it more reliable for tackling high-dimensional data.

4. Main Theoretical Result

In this section, we present the main theoretical results for our proposed algorithm. The following theorem shows the consistency of square loss, and the detailed proof is deferred in Section 5.1.

Theorem 1 *For square loss $\ell(t) = (1-t)^2$, the surrogate loss $\Psi(f, \mathbf{x}, \mathbf{x}') = \ell(f(\mathbf{x}) - f(\mathbf{x}'))$ is consistent with AUC.*

Define \mathbf{w}_* as

$$\mathbf{w}_* = \arg \min_{\mathbf{w}} \sum_t \mathcal{L}_t(\mathbf{w}).$$

We are in the position to present the following convergence rate for Algorithm 1 when the full covariance matrices are provided, and the detailed proof is deferred in Section 5.2

Theorem 2 *For $\|\mathbf{x}_t\| \leq 1$ ($t \in [T]$), $\|\mathbf{w}_*\| \leq B$ and $TL^* \geq \sum_{t=1}^T \mathcal{L}_t(\mathbf{w}_*)$, we have*

$$\sum_t \mathcal{L}_t(\mathbf{w}_t) - \sum_t \mathcal{L}_t(\mathbf{w}_*) \leq 2\kappa B^2 + B\sqrt{2\kappa TL^*},$$

where $\kappa = 4 + \lambda$ and $\eta_t = 1/(\kappa + \sqrt{(\kappa^2 + \kappa TL^*)/B^2})$.

This theorem presents an $O(1/T)$ convergence rate for the OPAUC algorithm if the distribution is separable, i.e., $L^* = 0$, and an $O(1/\sqrt{T})$ convergence rate for general case. Compared to the online AUC optimization algorithm (Zhao et al., 2011), which achieves at most $O(1/\sqrt{T})$ convergence rate, our proposed algorithm clearly reduce the regret. The faster convergence rate of our proposed algorithm owes to the smoothness of the square loss, an important property that has been explored by some studies of online learning (Rakhlin et al., 2012) and generalization error bound analysis (Srebro et al., 2010).

Remark: The bound in Theorem 2 does not explicitly explore the strongly convexity of $\mathcal{L}_t(\mathbf{w})$, which can lead to an $O(1/T)$ convergence rate. Instead, we focus on exploiting the smoothness of the loss function, since we did not introduce a bounded domain for \mathbf{w} . Due to the regularizer $\lambda|\mathbf{w}|^2/2$, we have $|\mathbf{w}_*| \leq 1/\lambda$, and it is reasonable to restrict \mathbf{w}_t by $|\mathbf{w}_t| \leq 1/\lambda$, leading to a regret bound of $O(\ln T/[\lambda^3 T])$ by applying the standard stochastic gradient descent with $\eta_t = 1/[\lambda t]$. This bound is preferred only when $\lambda = \Omega(T^{-1/6})$, a scenario which rarely occurs in empirical study. This problem may also be addressable by exploiting the epoch gradient method (Nocedal and Wright, 1999), a subject of our future study.

We now consider the case when covariance matrices are approximated by low-rank matrices. Note that the low-rank approximation is accurate only if the eigenvalues of covariance matrices follow a skewed distribution. To capture the skewed eigenvalue distribution, we introduce the concept of *effective numerical rank* (Hansen, 1987) that generalizes the rank of matrix:

Definition 3 For a positive constant $\mu > 0$ and semi-positive definite matrix $M \in \mathbb{R}^{d \times d}$ of eigenvalues $\{\nu_i\}$, the effective numerical rank w.r.t. μ is defined to be $r(M, \mu) = \sum_{i=1}^d \nu_i / (\mu + \nu_i)$.

It is evident that the effective numerical rank is upper bounded by the true rank, i.e., $r(M, \mu) \leq \text{rank}(M)$. To further see how the concept of effective numerical rank captures the skewed eigenvalue distribution, consider a PSD matrix M of full rank with $\sum_{i=k}^d \nu_i \leq \mu$ for small k . It is easy to verify that $r(M, \mu) \leq k$, i.e., M can be well approximated by a matrix of rank k .

Define the effective numerical rank for a set of matrices $\{M_t\}_{t=1}^T$ as

$$r(\{M_t\}_{t=1}^T, \mu) = \max_{1 \leq t \leq T} r(M_t, \mu).$$

Under the assumption that the effective numerical rank for the set of covariance matrices $\{S_t^\pm\}_{t=1}^T$ is small (i.e., S_t^\pm can be well approximated by low-rank matrices), the following theorem gives the convergence rate for $|\sum_t \hat{\mathcal{L}}_t(\mathbf{w}_t) - \sum_t \mathcal{L}_t(\mathbf{w}_*)|$, where $\hat{\mathcal{L}}_t(\mathbf{w}_t)$ are given by Eqs. (6) and (7).

Theorem 4 Let $r = r(\{S_t^\pm\}_{t=1}^T, \lambda)$ be the effective numerical rank for the sequence of covariance matrices $\{S_t^\pm\}_{t=1}^T$. For $0 < \delta < 1$, $0 < \epsilon \leq 1/2$, $|\mathbf{w}_*| \leq B$, $\|\mathbf{x}_t\| \leq 1$ ($t \in [T]$) and $TL^* \geq \sum_{t=1}^T \mathcal{L}_t(\mathbf{w}_*)$, we have with probability at least $1 - \delta$,

$$\left| \sum_t (\hat{\mathcal{L}}_t(\mathbf{w}_t) - \mathcal{L}_t(\mathbf{w}_*)) \right| \leq 2\epsilon TL^* + 2\kappa B^2 + B\sqrt{2\kappa TL^*}$$

provided $\tau \geq \frac{32r\lambda}{\epsilon^2} \log \frac{2dT}{\delta}$, where $\kappa = 4 + \lambda$ and $\eta_t = 1/(\kappa + \sqrt{(\kappa^2 + \kappa TL^*)/B^2})$.

The detailed proof is presented in Section 5.3. For the separable distribution $L^* = 0$, we also obtain an $O(1/T)$ convergence rate when the covariance matrices are approximated by low-rank matrices. Compared with Theorem 2, Theorem 4 introduces an additional term $2\epsilon L^*$ in the bound when using the approximate covariance matrices, and it is noteworthy that the approximation does not significantly increase the bound of Theorem 2 if $2\epsilon TL^* \leq B\sqrt{2(4+\lambda)TL^*}$, i.e., $\epsilon \leq B\sqrt{2(\lambda+4)/TL^*}$. This implies that the approximate algorithm will achieve similar performance as the one using the full covariance matrices provided $\tau = \Omega(r\lambda T(\log d + \log T)/(\lambda + 4))$. When $\lambda = O(1/T)$, this requirement is reduced to $\tau = \Omega(r[\log d + \log T])$, a logarithmic dependence on dimension d .

5. Proofs

In this section, we present detailed proofs for our main theorems.

5.1 Proof of Theorem 1

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ with instance-marginal probability $p_i = \Pr[x_i]$ and conditional probability $\xi_i = \Pr[y = +1|\mathbf{x}_i]$, and we denote by the expected risk

$$R_\Psi(f) = C_0 + \sum_{i \neq j} p_i p_j (\xi_i(1 - \xi_j)\ell(f(\mathbf{x}_i) - f(\mathbf{x}_j)) + \xi_j(1 - \xi_i)\ell(f(\mathbf{x}_j) - f(\mathbf{x}_i)))$$

where $\ell(t) = (1 - t)^2$ and C_0 is a constant with respect to $f(\mathbf{x}_i)$ ($1 \leq i \leq n$). According to the analysis of (Gao and Zhou, 2012), it suffices to prove that, for every optimal solution f , i.e., $R_\Psi(f) = \inf_{f'} R_\Psi(f')$, we have $f(\mathbf{x}_i) > f(\mathbf{x}_j)$ if $\xi_i > \xi_j$.

If $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2\}$, then minimizing $R_\phi(f)$ gives the optimal solution $f = (f(\mathbf{x}_1), f(\mathbf{x}_2))$ such that

$$f(\mathbf{x}_1) - f(\mathbf{x}_2) = \text{sgn}(\xi(\mathbf{x}_1) - \xi(\mathbf{x}_2)) \text{ for } \xi(\mathbf{x}_1) \neq \xi(\mathbf{x}_2),$$

which shows the consistency of least square loss.

For $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ with $n \geq 3$, if $\xi_i(1 - \xi_i) = 0$ for every $1 \leq i \leq n$, then minimizing $R_\Psi(f)$ gives the optimal solution $f = (f_1, f_2, \dots, f_n)$ such that

$$f_j = f_i + 1 \text{ for every } \xi_i = 1 \text{ and } \xi_j = -1,$$

which shows the consistency of least square loss.

If $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ with $n \geq 3$, and there exists some i_0 s.t. $\xi_{i_0}(1 - \xi_{i_0}) \neq 0$, then the subgradient conditions give optimal solution such that

$$\sum_{k \neq i} p_k (\xi_i + \xi_k - 2\xi_i \xi_k) (f(\mathbf{x}_i) - f(\mathbf{x}_k)) = \sum_{k \neq i} p_k (\xi_i - \xi_k) \text{ for each } 1 \leq i \leq n.$$

Solving the above n linear equations, we obtain the optimal solution $f = (f_1, f_2, \dots, f_n)$, i.e.,

$$f(\mathbf{x}_i) - f(\mathbf{x}_j) = (\xi_i - \xi_j) \frac{\prod_{k \neq i, j} \sum_{l=1}^n p_l (\xi_l + \xi_k - 2\xi_l \xi_k)}{\sum_{\substack{s_i \geq 0 \\ s_1 + \dots + s_n = n-2}} p_1^{s_1} \dots p_n^{s_n} \Gamma(s_1, s_2, \dots, s_n)}$$

where Γ is a polynomial in $\xi[k_1] + \xi[k_2] - 2\xi[k_1]\xi[k_2]$ for $1 \leq k_1, k_2 \leq n$. In the following, we will derive the specific expression for $\Gamma(s_1, s_2, \dots, s_n)$. Denote by $\mathcal{A} = \{i: s_i \geq 1\}$ and $\mathcal{B} = \{i: s_i = 0\} = \{b_1, b_2, \dots, b_{|\mathcal{B}|}\}$.

- If $|\mathcal{A}| = 1$, i.e., $\mathcal{A} = \{i_1\}$ for some $1 \leq i_1 \leq n$, then

$$\Gamma(s_1, s_2, \dots, s_n) = \prod_{k \in \mathcal{B}} (\xi_{i_1} + \xi_k - 2\xi_{i_1}\xi_k).$$

- If $|\mathcal{A}| = 2$, i.e., $\mathcal{A} = \{i_1, i_2\}$ for some $1 \leq i_1, i_2 \leq n$, then we denote by

$$\mathcal{A}_1 = \{s_{i_1} \odot i_1\} \cup \{s_{i_2} \odot i_2\}$$

where $\{s_{i_k} \odot i_k\}$ denotes the multi-set $\{i_k, i_k, \dots, i_k\}$ of size s_{i_k} for $k = 1, 2$. It is clear that $|\mathcal{B}| = |\mathcal{A}_1| = n - 2$. Further, we denote by $\mathcal{G}(\mathcal{A}_1)$ the set of all permutations of \mathcal{A}_1 . Therefore, we have

$$\Gamma(s_1, s_2, \dots, s_n) = (\xi_{i_1} + \xi_{i_2} - 2\xi_{i_1}\xi_{i_2}) \sum_{\pi=\pi_1 \dots \pi_{n-2} \in \mathcal{G}(\mathcal{A}_1)} \prod_{k=1}^{n-2} (\xi_{\pi_k} + \xi_{b_k} - 2\xi_{\pi_k}\xi_{b_k}).$$

- If $|\mathcal{A}| > 2$, then, for $i_1 \neq i_2 \in \mathcal{A}$, we denote by the multi-set

$$\mathcal{A}_1(i_1, i_2) = \{s_{i_1} \odot i_1\} \cup \{s_{i_2} \odot i_2\} \cup \left(\bigcup_{k \in \mathcal{A} \setminus \{i_1, i_2\}} \{(s_k - 1) \odot k\} \right),$$

and it is easy to derive $|\mathcal{A}_1| = |\mathcal{B}|$. Further, we denote by $\mathcal{G}(\mathcal{A} \setminus \{i_1, i_2\})$ and $\mathcal{G}(\mathcal{A}_1)$ the set of all permutations of $\mathcal{A} \setminus \{i_1, i_2\}$ and \mathcal{A}_1 , respectively. Therefore, we set

$$\begin{aligned} \Gamma_1(i_1, i_2, \mathcal{A}) &= \sum_{\pi=\pi_1 \pi_2 \dots \pi_{|\mathcal{A}|-2} \in \mathcal{G}(\mathcal{A} \setminus \{i_1, i_2\})} (\xi_{i_1} + \xi_{\pi_1} - 2\xi_{i_1}\xi_{\pi_1})(\xi_{\pi_1} + \xi_{\pi_2} - 2\xi_{\pi_1}\xi_{\pi_2}) \\ &\quad \times \dots \times (\xi_{\pi_{|\mathcal{A}|-3}} + \xi_{\pi_{|\mathcal{A}|-2}} - 2\xi_{\pi_{|\mathcal{A}|-3}}\xi_{\pi_{|\mathcal{A}|-2}})(\xi_{\pi_{|\mathcal{A}|-2}} + \xi_{i_2} - 2\xi_{i_2}\xi_{\pi_{|\mathcal{A}|-2}}), \end{aligned}$$

and we have

$$\Gamma(s_1, s_2, \dots, s_n) = \sum_{\substack{i_1 \neq i_2 \\ i_1, i_2 \in \mathcal{A}}} \Gamma_1(i_1, i_2, \mathcal{A}) \sum_{\pi=\pi_1 \pi_2 \dots \pi_{|\mathcal{B}|} \in \mathcal{G}(\mathcal{A}_1)} \prod_{k=1}^{|\mathcal{B}|} (\xi_{\pi_k} + \xi_{b_k} - 2\xi_{\pi_k}\xi_{b_k})$$

where $\mathcal{B} = \{b_1, b_2, \dots, b_{|\mathcal{B}|}\}$.

Since there exist some i_0 s.t. $\xi_{i_0}(1 - \xi_{i_0}) \neq 0$, we have

$$\frac{\prod_{k \neq i, j} \sum_{l=1}^n p_l (\xi_l + \xi_k - 2\xi_l \xi_k)}{\sum_{\substack{s_i \geq 0 \\ s_1 + \dots + s_n = n-2}} p_1^{s_1} \dots p_n^{s_n} \Gamma(s_1, s_2, \dots, s_n)} > 0.$$

Therefore, it is evident that $f(\mathbf{x}_i) > f(\mathbf{x}_j)$ if $\xi_i > \xi_j$, and this theorem follows as desired. ■

5.2 Proof of Theorem 2

This proof is motivated from (Shalev-Shwartz, 2007; Srebro et al., 2010). Recall

$$\mathcal{L}_t(\mathbf{w}) = \frac{\lambda}{2}|\mathbf{w}|^2 + \frac{\sum_{i=1}^{t-1} \mathbb{I}[y_i \neq y_t](1 - y_t(\mathbf{x}_t - \mathbf{x}_i)^\top \mathbf{w})^2}{2|\{i \in [t-1] : y_i \neq y_t\}|}.$$

For $|\mathbf{w}_*| \leq B$ and convex $\mathcal{L}_t(\mathbf{w})$, we have

$$\mathcal{L}_t(\mathbf{w}_t) - \mathcal{L}_t(\mathbf{w}_*) \leq \nabla \mathcal{L}_t(\mathbf{w}_t)^\top (\mathbf{w}_t - \mathbf{w}_*). \quad (12)$$

It is easy to derive that $\nabla \mathcal{L}_t(\mathbf{w}_t)$ equals to

$$\lambda \mathbf{w}_t - \frac{\sum_{i=1}^{t-1} \mathbb{I}[y_i \neq y_t](1 - y_t(\mathbf{x}_t - \mathbf{x}_i)^\top \mathbf{w}_t)y_t(\mathbf{x}_t - \mathbf{x}_i)}{|\{i \in [t-1] : y_i \neq y_t\}|},$$

and therefore, for any \mathbf{w} and $|\mathbf{x}_i| \leq 1$

$$|\nabla \mathcal{L}_t(\mathbf{w}_t) - \nabla \mathcal{L}_t(\mathbf{w})| \leq (4 + \lambda)|\mathbf{w}_t - \mathbf{w}|.$$

Denote by

$$\mathbf{w}_{t*} = \arg \min_{\mathbf{w}} \mathcal{L}_t(\mathbf{w}_t),$$

which implies that $\nabla \mathcal{L}_t(\mathbf{w}_{t*}) = 0$ for convex and smooth \mathcal{L}_t . Based on (Nesterov, 2003, Theorem 2.1.5), we have

$$|\nabla \mathcal{L}_t(\mathbf{w}_t)|^2 = |\nabla \mathcal{L}_t(\mathbf{w}_t) - \nabla \mathcal{L}_t(\mathbf{w}_{t*})|^2 \leq 2(\lambda + 4)L_t(w_t) \quad (13)$$

where the inequality holds from $\mathcal{L}_t(\mathbf{w}_{t*}) \geq 0$ and $\nabla \mathcal{L}_t(\mathbf{w}_{t*}) = 0$. Moreover, we have

$$|\mathbf{w}_{t+1} - \mathbf{w}_*|^2 = |\mathbf{w}_t - \eta_t \nabla \mathcal{L}_t(\mathbf{w}_t) - \mathbf{w}_*|^2 = |\mathbf{w}_t - \mathbf{w}_*|^2 - 2\eta_t \nabla \mathcal{L}_t(\mathbf{w}_t)^\top (\mathbf{w}_t - \mathbf{w}_*) + \eta_t^2 |\nabla \mathcal{L}_t(\mathbf{w}_t)|^2,$$

and this yields that, by using Eqs. (12) and (13),

$$(1 - (4 + \lambda)\eta_t)\mathcal{L}_t(\mathbf{w}_t) - \mathcal{L}_t(\mathbf{w}_*) \leq \frac{1}{2\eta_t}|\mathbf{w}_t - \mathbf{w}_*|^2 - \frac{1}{2\eta_t}|\mathbf{w}_{t+1} - \mathbf{w}_*|^2.$$

Summing over $t = 0, \dots, T-1$ and rearranging, we obtain

$$\begin{aligned} & \sum_{t=0}^{T-1} (1 - (4 + \lambda)\eta_t)\mathcal{L}_t(\mathbf{w}_t) - \sum_{t=0}^{T-1} \mathcal{L}_t(\mathbf{w}_*) \\ & \leq \frac{1}{2\eta_0}|\mathbf{w}_0 - \mathbf{w}_*|^2 - \frac{1}{2\eta_{T-1}}|\mathbf{w}_T - \mathbf{w}_*|^2 + \sum_{i=1}^{T-2} \left(\frac{1}{2\eta_{i+1}} - \frac{1}{2\eta_i}\right)|\mathbf{w}_i - \mathbf{w}_*|^2. \end{aligned}$$

By setting $\eta_t = \eta$, we have

$$\frac{1}{2\eta_0}|\mathbf{w}_0 - \mathbf{w}_*|^2 - \frac{1}{2\eta_{T-1}}|\mathbf{w}_T - \mathbf{w}_*|^2 \leq \frac{1}{2\eta}|\mathbf{w}_*|^2 \leq \frac{B^2}{2\eta}$$

from $\mathbf{w}_0 = \mathbf{0}$ and $|\mathbf{w}_*| \leq B$, and we further get

$$\begin{aligned} \sum_{t=0}^{T-1} \mathcal{L}_t(\mathbf{w}_t) - \sum_{t=0}^{T-1} \mathcal{L}_t(\mathbf{w}_*) &\leq \frac{1}{1 - (4 + \lambda)\eta} \left(\frac{B^2}{2\eta} + (4 + \lambda)\eta \sum_{t=0}^{T-1} \mathcal{L}_t(\mathbf{w}_*) \right) \\ &\leq \frac{1}{1 - (4 + \lambda)\eta} \left(\frac{B^2}{2\eta} + (4 + \lambda)\eta TL^* \right). \end{aligned}$$

This theorem holds by putting

$$\eta = \frac{1}{4 + \lambda + \sqrt{(4 + \lambda)^2 + (4 + \lambda)TL^*/B^2}}$$

into the above formula and using the formula $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. \blacksquare

5.3 Proof of Theorem 4

Before the detailed proof of Theorem 4, we begin with some useful results:

Lemma 5 *Let $S_1 = \text{diag}(s_{1i})$ and $S_2 = \text{diag}(s_{2i})$ be two $d \times d$ diagonal matrices such that $s_{1i} \neq 0$ and $s_{1i}^2 + s_{2i}^2 = 1$ for all i . For a Gaussian random matrix $R \in \mathbb{R}^{d \times \tau}$, we set $Z = S_1 S_1 + S_2 R R^\top S_2$ and $r = \sum_i s_{2i}^2$, and the followings hold*

$$\Pr[\lambda_1(Z) \geq 1 + \epsilon] \leq d \exp(-\tau \epsilon^2 / 32r) \quad \text{and} \quad \Pr[\lambda_p(Z) \leq 1 - \epsilon] \leq d \exp(-\tau \epsilon^2 / 32r),$$

where $\lambda_k(Z)$ denotes the k -th largest eigenvalue of matrix Z .

Proof This proof technique is motivated from (Gittens and Tropp, 2011) by adding a bias matrix. Let $g(\theta) = \frac{\theta^2}{2-2\theta}$. Then, we have

$$\begin{aligned} \Pr[\lambda_1(M) \geq 1 + \epsilon] &\leq \inf_{\theta > 0} \Pr \left\{ \theta \left(S_1 S_1 + E[S_2 R R^\top S_2] - (1 + \epsilon)I \right) + g(\theta) E[(S_2 R R^\top S_2)^2] / \tau \right\} \\ &\leq \inf_{\theta > 0} \Pr \left\{ -\theta \epsilon + 8g(\theta) \text{tr}(S_2^2) S_2^2 \right\} \\ &\leq \inf_{\theta > 0} d \exp\{-\theta \epsilon + 8r g(\theta)\} \leq d \exp(-\tau \epsilon^2 / 32r). \end{aligned}$$

In a similar manner,

$$\begin{aligned} \Pr[\lambda_p(M) \leq 1 - \epsilon] &\leq \inf_{\theta > 0} \Pr \left\{ \theta (S_1 S_1 + E[S_2 R R^\top S_2] - (1 - \epsilon)I) + g(\theta) E[(S_2 R R^\top S_2)^2] / \tau \right\} \\ &\leq \inf_{\theta > 0} d \exp\{-\theta \epsilon + 8r g(\theta)\} \leq d \exp(-\tau \epsilon^2 / 32r). \end{aligned}$$

This completes the proof. \blacksquare

Let $M \in \mathbb{R}^{d \times d}$ be a positive semi-definite (PSD) matrix with effective numerical rank $r(M, \mu)$ for $\mu > 0$. We define two matrices K and \hat{K} , respectively, as

$$K := \mu I_d + M \quad \text{and} \quad \hat{K} := \mu I_d + M^{-1/2} R R^\top M^{-1/2},$$

where $R \in \mathbb{R}^{d \times m}$ is a (Gaussian) random matrix. Based on Lemma 5, we have the following theorem that bounds the difference between $K - \hat{K}$:

Lemma 6 *Let $r(M, \mu)$ be the numerical rank for $\mu > 0$ and PSD matrix M . Then, for $\delta > 0$ and $\epsilon > 0$, the following holds with probability at least $1 - \delta$*

$$\|I - K^{-1/2} \hat{K} K^{-1/2}\|_2 \leq \epsilon,$$

where $\|Z\|_2$ measures the spectral norm of matrix Z , provided

$$\tau \geq \frac{32r(M, \mu)}{\epsilon^2} \log(2d/\delta).$$

Proof Let $M = U \text{diag}(\sigma_i^2) V^\top$ be the singular value decomposition of M . We define

$$S_1 = \text{diag} \left(\sqrt{\frac{\mu}{\sigma_1^2 + \mu}}, \dots, \sqrt{\frac{\mu}{\sigma_d^2 + \mu}} \right) \quad \text{and} \quad S_2 = \text{diag} \left(\frac{\sigma_1}{\sqrt{\sigma_1^2 + \mu}}, \dots, \frac{\sigma_d}{\sqrt{\sigma_d^2 + \mu}} \right).$$

It is easy to observe that

$$Z = K^{-1/2} \hat{K} K^{-1/2} = U(S_1^2 + S_2 V^\top R R^\top V) U^\top = U(S_1^2 + S_2 \hat{R} \hat{R}^\top) U^\top$$

where $\hat{R} = V^\top R \in \mathbb{R}^{d \times \tau}$ is also Gaussian random matrix because V is an orthonormal matrix. Parameter r in Lemma 5 is given by

$$r = \sum_{i=1}^d s_{2,i}^2 = \sum_{i=1}^d \frac{\sigma_i^2}{\sigma_i^2 + \mu} = r(M, \mu).$$

Using Lemma 5, the followings hold with a probability at least $1 - \delta$,

$$\lambda_{\max}(Z) = \|K^{-1/2} \hat{K} K^{-1/2}\|_2 \leq 1 + \epsilon \quad \text{and} \quad \lambda_{\min}(Z) = \lambda_d(K^{-1/2} \hat{K} K^{-1/2}) \geq 1 - \epsilon,$$

which yields that $\|Z - I\| \leq \epsilon$ provided

$$\tau \geq \frac{32r}{\epsilon^2} \log \frac{2d}{\delta}.$$

This lemma follows as desired. ■

Recall that

$$\hat{\mathcal{L}}_t(\mathbf{w}) = \frac{\lambda}{2} |\mathbf{w}|^2 + \mathbf{w}^\top (\mathbf{c}_{t-1}^- - \mathbf{x}_t) + \frac{1}{2} + \frac{1}{2} \left(\mathbf{w}^\top (\mathbf{x}_t - \mathbf{c}_{t-1}^-) (\mathbf{x}_t - \mathbf{c}_{t-1}^-)^\top \mathbf{w} + \mathbf{w}^\top \hat{S}_t^- \mathbf{w} \right)$$

if $y_t = 1$; otherwise,

$$\hat{\mathcal{L}}_t(\mathbf{w}) = \frac{\lambda}{2} |\mathbf{w}|^2 + \mathbf{w}^\top (\mathbf{x}_t - \mathbf{c}_{t-1}^+) + \frac{1}{2} + \frac{1}{2} \left(\mathbf{w}^\top (\mathbf{x}_t - \mathbf{c}_{t-1}^+) (\mathbf{x}_t - \mathbf{c}_{t-1}^+)^\top \mathbf{w} + \mathbf{w}^\top \hat{S}_t^+ \mathbf{w} \right).$$

We further define $\hat{\mathbf{w}}_*$ as the optimal solution that minimizes the loss based on approximate covariance matrices, i.e.

$$\hat{\mathbf{w}}_* = \arg \min_{\mathbf{w}} \sum_{t=1}^T \hat{\mathcal{L}}_t(\mathbf{w}).$$

Based on Lemma 6, the following theorem gives an upper bound for $|\sum_t \hat{\mathcal{L}}_t(\hat{\mathbf{w}}_*) - \sum_t \mathcal{L}_t(\mathbf{w}_*)|$.

Theorem 7 Let $r(\{S_t^\pm\}_{t=1}^T, \lambda)$ be the effective numerical rank for the set of covariance matrices $S_t^\pm, t = 1, \dots, T$ with respect to the regularization parameter λ . Then, for any $0 < \delta$ and $0 < \epsilon \leq 1/2$, the followings hold with probability at least $1 - \delta$

$$|\hat{\mathbf{w}}_* - \mathbf{w}_*| \leq 2\epsilon |\mathbf{w}_*| \quad (14)$$

$$\left| \sum_t \hat{\mathcal{L}}_t(\hat{\mathbf{w}}_*) - \sum_t \mathcal{L}_t(\mathbf{w}_*) \right| \leq 2\epsilon \sum_t \mathcal{L}_t(\mathbf{w}_*) \quad (15)$$

provided that

$$\tau \geq \frac{32r(S, \lambda)}{\epsilon^2} \log \frac{2d}{\delta T}$$

Proof We first rewrite $\mathcal{L}(\mathbf{w}) = \sum_{t=1}^T \mathcal{L}_t(\mathbf{w})$ as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} + \mathbf{w}^\top \mathbf{a} + \frac{1}{2} \mathbf{w}^\top (A_1 + A_2) \mathbf{w}$$

where

$$\begin{aligned} \mathbf{a} &= \sum_{t=1}^T \frac{\mathbb{I}[y_t = 1] (\mathbf{c}_{t-1}^- - \mathbf{x}_t) + \mathbb{I}[y_t = -1] (\mathbf{c}_{t-1}^+ - \mathbf{x}_t)}{T} \\ A_1 &= \lambda I_d + \frac{1}{T} \sum_{t=1}^T (\mathbb{I}[y_t = 1] S_{t-1}^- + \mathbb{I}[y_t = -1] S_{t-1}^+) \\ A_2 &= \frac{1}{T} \sum_{t=1}^T \mathbb{I}[y_t = 1] (\mathbf{x}_t - \mathbf{c}_t^-)(\mathbf{x}_t - \mathbf{c}_t^-)^\top + \frac{1}{T} \sum_{t=1}^T \mathbb{I}[y_t = -1] (\mathbf{x}_t - \mathbf{c}_t^+)(\mathbf{x}_t - \mathbf{c}_t^+)^\top. \end{aligned}$$

Similarly, we rewrite $\hat{\mathcal{L}}(\mathbf{w}) = \sum_{t=1}^T \hat{\mathcal{L}}_t(\mathbf{w})$ as

$$\hat{\mathcal{L}}(\mathbf{w}) = \frac{1}{2} + \mathbf{w}^\top \mathbf{a} + \frac{1}{2} \mathbf{w}^\top (\tilde{A}_1 + A_2) \mathbf{w}$$

where

$$\tilde{A}_1 = \lambda I_d + \frac{1}{T} \sum_{t=1}^T \mathbb{I}[y_t = 1] [S_{t-1}^-]^{1/2} R_t R_t^\top [S_{t-1}^-]^{1/2} + \frac{1}{T} \sum_{t=1}^T \mathbb{I}[y_t = -1] [S_{t-1}^+]^{1/2} R_t R_t^\top [S_{t-1}^+]^{1/2}.$$

The optimal solutions for minimizing $\mathcal{L}(\mathbf{w})$ and $\hat{\mathcal{L}}(\mathbf{w})$ are given, respectively, by

$$\mathbf{w}_* = (A_1 + A_2)^{-1} \mathbf{a} \quad \text{and} \quad \hat{\mathbf{w}}_* = (\tilde{A}_1 + A_2)^{-1} \mathbf{a}.$$

Define $\Delta = I - A_1^{-1/2} \tilde{A}_1 A_1^{-1/2}$ and write \tilde{A}_1 in terms of Δ as

$$\tilde{A}_1 = A_1 - A_1^{1/2} \Delta A_1^{1/2}$$

Using Lemma 6, it holds that $\epsilon I_d \preceq \Delta \preceq \epsilon I_d$ with probability at least $1 - \delta T$, and therefore

$$(1 - \epsilon)(A_1 + A_2) \preceq \tilde{A}_1 + A_2 \preceq (1 + \epsilon)(A_1 + A_2)$$

Table 1: Benchmark datasets

datasets	#inst	#feat	datasets	#inst	#feat
diabetes	768	8	w8a	49,749	300
fourclass	862	2	kddcup04	50,000	65
german	1,000	24	mnist	60,000	780
splice	3,175	60	connect-4	67,557	126
usps	9,298	256	acoustic	78,823	50
letter	15,000	16	ijcnn1	141,691	22
magic04	19,020	10	epsilon	400,000	2,000
a9a	32,561	123	covtype	581,012	54

Denote by

$$\Omega = (A_1 + A_2)^{1/2}(A_1 + A_2)^{-1}(A_1 + A_2)^{1/2} - I,$$

and according to previous analysis, we have

$$|\Omega| \leq \frac{\epsilon}{1 - \epsilon} \leq 2\epsilon \quad (16)$$

for $\epsilon < 1/2$. Therefore,

$$\begin{aligned} |\hat{\mathbf{w}}_* - \mathbf{w}_*| &= \left| \left((\tilde{A}_1 + A_2)^{-1} - (A_1 + A_2)^{-1} \right) \mathbf{a} \right| \\ &= |(A_1 + A_2)^{-1/2} \Omega (A_1 + A_2)^{-1/2} \mathbf{a}| \\ &\leq 2\epsilon |(A_1 + A_2)^{-1} \mathbf{a}| \leq 2\epsilon |\mathbf{w}_*| \end{aligned}$$

and

$$\begin{aligned} \left| \hat{\mathcal{L}}(\mathbf{w}_*) - \mathcal{L}(\mathbf{w}_*) \right| &= \frac{3}{2} \left| \mathbf{a}^\top \left((\tilde{A}_1 + A_2)^{-1} - (A_1 + A_2)^{-1} \right) \mathbf{a} \right| \\ &= \frac{3}{2} \left| \mathbf{a}^\top \left((A_1 + A_2)^{-1/2} \Omega (A_1 + A_2)^{-1/2} \right) \mathbf{a} \right| \\ &\leq 2\epsilon \left| \frac{3}{2} \mathbf{a}^\top (A_1 + A_2)^{-1} \mathbf{a} \right| \leq 2\epsilon \left| \frac{1}{2} + \frac{3}{2} \mathbf{a}^\top (A_1 + A_2)^{-1} \mathbf{a} \right| = 2\epsilon \mathcal{L}(\mathbf{w}_*). \end{aligned}$$

This theorem follows as desired. ■

Proof of Theorem 4: For $|w_*| \leq B$, $\mathcal{L}(\mathbf{w}_*) \leq TL^*$, and $0 < \epsilon \leq 1/2$, we have

$$|\hat{\mathbf{w}}_*| \leq |\mathbf{w}_*| + |\hat{\mathbf{w}}_* - \mathbf{w}_*| \leq 2B \quad (17)$$

from Eq. (14), and we further have

$$\hat{\mathcal{L}}^*(\hat{\mathbf{w}}_*) \leq \mathcal{L}(\mathbf{w}_*) + |\hat{\mathcal{L}}^*(\hat{\mathbf{w}}_*) - \mathcal{L}(\mathbf{w}_*)| \leq 2\mathcal{L}(\mathbf{w}_*) \leq 2TL^* \quad (18)$$

from Eq. (15). Therefore, we have

$$\left| \sum_t \hat{\mathcal{L}}(\mathbf{w}_t) - \sum_t \mathcal{L}(\mathbf{w}_*) \right| \leq \sum_t \hat{\mathcal{L}}(\mathbf{w}_t) - \sum_t \mathcal{L}(\hat{\mathbf{w}}_*) + \left| \sum_t \hat{\mathcal{L}}(\hat{\mathbf{w}}_*) - \sum_t \mathcal{L}(\mathbf{w}_*) \right|. \quad (19)$$

Table 2: Testing AUC (mean \pm std.) of OPAUC with online algorithms on benchmark datasets. \bullet/\circ indicates that OPAUC is significantly better/worse than the corresponding method (pairwise t -tests at 95% significance level).

datasets	OPAUC	OAM _{seq}	OAM _{gra}	online Uni-Exp	online Uni-Squ
diabetes	.8309 \pm .0350	.8264 \pm .0367	.8262 \pm .0338	.8215 \pm .0309 \bullet	.8258 \pm .0354
fourclass	.8310 \pm .0251	.8306 \pm .0247	.8295 \pm .0251	.8281 \pm .0305	.8292 \pm .0304
german	.7978 \pm .0347	.7747 \pm .0411 \bullet	.7723 \pm .0358 \bullet	.7908 \pm .0367	.7899 \pm .0349
splice	.9232 \pm .0099	.8594 \pm .0194 \bullet	.8864 \pm .0166 \bullet	.8931 \pm .0213 \bullet	.9153 \pm .0132 \bullet
usps	.9620 \pm .0040	.9310 \pm .0159 \bullet	.9348 \pm .0122 \bullet	.9538 \pm .0045 \bullet	.9563 \pm .0041 \bullet
letter	.8114 \pm .0065	.7549 \pm .0344 \bullet	.7603 \pm .0346 \bullet	.8113 \pm .0074	.8053 \pm .0081 \bullet
magic04	.8383 \pm .0077	.8238 \pm .0146 \bullet	.8259 \pm .0169 \bullet	.8354 \pm .0099 \bullet	.8344 \pm .0086 \bullet
a9a	.9002 \pm .0047	.8420 \pm .0174 \bullet	.8571 \pm .0173 \bullet	.9005 \pm .0024	.8949 \pm .0025 \bullet
w8a	.9633 \pm .0035	.9304 \pm .0074 \bullet	.9418 \pm .0070 \bullet	.7693 \pm .0986 \bullet	.8847 \pm .0130 \bullet
kddcup04	.7912 \pm .0039	.6918 \pm .0412 \bullet	.7097 \pm .0420 \bullet	.7851 \pm .0050 \bullet	.7850 \pm .0042 \bullet
mnist	.9242 \pm .0021	.8615 \pm .0087 \bullet	.8643 \pm .0112 \bullet	.7932 \pm .0245 \bullet	.9156 \pm .0027 \bullet
connect-4	.8760 \pm .0023	.7807 \pm .0258 \bullet	.8128 \pm .0230 \bullet	.8702 \pm .0025 \bullet	.8685 \pm .0033 \bullet
acoustic	.8192 \pm .0032	.7113 \pm .0590 \bullet	.7711 \pm .0217 \bullet	.8171 \pm .0034 \bullet	.8193 \pm .0035
ijcnn1	.9269 \pm .0021	.9209 \pm .0079 \bullet	.9100 \pm .0092 \bullet	.9264 \pm .0035	.9022 \pm .0041 \bullet
epsilon	.9550 \pm .0007	.8816 \pm .0042 \bullet	.8659 \pm .0176 \bullet	.9488 \pm .0012 \bullet	.9480 \pm .0021 \bullet
covtype	.8244 \pm .0014	.7361 \pm .0317 \bullet	.7403 \pm .0289 \bullet	.8236 \pm .0017	.8236 \pm .0020
win/tie/loss		14/2/0	14/2/0	10/6/0	11/5/0

We use Theorem 1 (in the main paper) to bound the first term in the above by combining Eqs. (17) and (18), and the second term can be bounded by Eq. (15). This completes the proof as desired. \blacksquare

6. Experiments

We evaluate the performance of OPAUC on benchmark datasets and high-dimensional datasets in Sections 6.1 and 6.2, respectively. Then, we study the parameter influence in Section 6.3.

6.1 Comparison on Benchmark Data

We conduct our experiments on sixteen benchmark datasets^{1,2,3} as summarized in Table 1. Some datasets have been used in previous studies on AUC optimization, whereas the other are large ones requiring one-pass procedure. The features have been scaled to $[-1, 1]$ for all datasets. Multi-class datasets have been transformed into binary ones by randomly partitioning classes into two groups, where each group contains the same number of classes.

1. <http://www.sigkdd.org/kddcup/>

2. <http://www.ics.uci.edu/~mlearn/MLRepository.html>

3. <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/>

Table 3: Testing AUC (mean \pm std.) of OPAUC with batch algorithms on benchmark datasets. \bullet/\circ indicates that OPAUC is significantly better/worse than the corresponding method (pairwise t -tests at 95% significance level).

datasets	OPAUC	SVM-perf	batch SVM-OR	batch LS-SVM	batch Uni-Log	batch Uni-Squ
diabetes	.8309 \pm .0350	.8325 \pm .0220	.8326 \pm .0328	.8325 \pm .0329	.8330 \pm .0322	.8332 \pm .0323
fourclass	.8310 \pm .0251	.8221 \pm .0381	.8305 \pm .0311	.8309 \pm .0309	.8288 \pm .0307	.8297 \pm .0310
german	.7978 \pm .0347	.7952 \pm .0340	.7935 \pm .0348	.7994 \pm .0343	.7995 \pm .0344	.7990 \pm .0342
splice	.9232 \pm .0099	.9235 \pm .0091	.9239 \pm .0089	.9245 \pm .0092 \circ	.9208 \pm .0107 \bullet	.9211 \pm .0107 \bullet
usps	.9620 \pm .0040	.9600 \pm .0054 \bullet	.9630 \pm .0047 \circ	.9634 \pm .0045 \circ	.9637 \pm .0041 \circ	.9617 \pm .0043
letter	.8114 \pm .0065	.8028 \pm .0074 \bullet	.8144 \pm .0064 \circ	.8124 \pm .0065 \circ	.8121 \pm .0061	.8112 \pm .0061
magic04	.8383 \pm .0077	.8427 \pm .0078 \circ	.8426 \pm .0074 \circ	.8379 \pm .0078	.8378 \pm .0073	.8338 \pm .0073 \bullet
a9a	.9002 \pm .0047	.9033 \pm .0039	.9009 \pm .0036	.8982 \pm .0028 \bullet	.9033 \pm .0025 \circ	.8967 \pm .0028 \bullet
w8a	.9633 \pm .0035	.9626 \pm .0042	.9495 \pm .0082 \bullet	.9495 \pm .0092 \bullet	.9421 \pm .0062 \bullet	.9075 \pm .0104 \bullet
kddcup04	.7912 \pm .0039	.7935 \pm .0037 \circ	.7903 \pm .0039 \bullet	.7898 \pm .0039 \bullet	.7900 \pm .0039 \bullet	.7926 \pm .0038
mnist	.9242 \pm .0021	.9338 \pm .0022 \circ	.9340 \pm .0020 \circ	.9336 \pm .0025 \circ	.9334 \pm .0021 \circ	.9279 \pm .0021 \circ
connect-4	.8760 \pm .0023	.8794 \pm .0024 \circ	.8749 \pm .0025 \bullet	.8739 \pm .0026 \bullet	.8784 \pm .0026 \circ	.8760 \pm .0024
acoustic	.8192 \pm .0032	.8102 \pm .0032 \bullet	.8262 \pm .0032 \circ	.8210 \pm .0033 \circ	.8253 \pm .0032 \circ	.8222 \pm .0031 \circ
ijcnn1	.9269 \pm .0021	.9314 \pm .0025 \circ	.9337 \pm .0024 \circ	.9320 \pm .0037 \circ	.9282 \pm .0023 \circ	.9038 \pm .0025 \bullet
epsilon	.9550 \pm .0007	.8640 \pm .0049 \bullet	.8643 \pm .0053 \bullet	.8644 \pm .0050 \bullet	.8647 \pm .0150 \bullet	.8653 \pm .0073 \bullet
covtype	.8244 \pm .0014	.8271 \pm .0011 \circ	.8248 \pm .0013	.8222 \pm .0014 \bullet	.8246 \pm .0010	.8242 \pm .0012
win/tie/loss		4/6/6	4/6/6	6/4/6	4/6/6	6/8/2

In addition to state-of-the-art online AUC approaches $\mathbf{OAM}_{\text{seq}}$ and $\mathbf{OAM}_{\text{gra}}$ (Zhao et al., 2011), we also compare with:

- **online Uni-Exp**: An online learning algorithm which optimizes the (weighted) univariate exponential loss (Kotlowski et al., 2011);
- **online Uni-Squ**: An online learning algorithm which optimizes the (weighted) univariate square loss;
- **SVM-perf**: A batch learning algorithm which directly optimizes AUC (Joachims, 2005);
- **batch SVM-OR**: A batch learning algorithm which optimizes the pairwise hinge loss (Joachims, 2006);
- **batch LS-SVM**: A batch learning algorithm which optimizes the pairwise square loss;
- **batch Uni-Log**: A batch learning algorithm which optimizes the (weighted) univariate logistic loss (Kotlowski et al., 2011);
- **batch Uni-Squ**: A batch learning algorithm which optimizes the (weighted) univariate square loss.

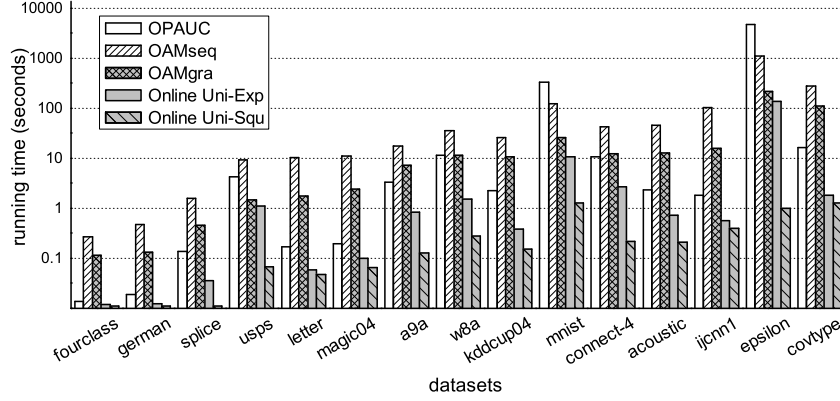


Figure 1: Comparison of the running time (in seconds) of OPAUC and online learning algorithms on benchmark data sets. Notice that the y -axis is in log-scale.

Table 4: High-dimensional datasets

datasets	#inst	#feat	datasets	#inst	#feat
sector	9,619	55,197	news20.binary	19,996	1,355,191
sector.lvr	9,619	55,197	rcv1v2	23,149	47,236
news20	15,935	62,061	ecml2012	456,886	98,519

All experiments are performed with Matlab 7 on a node of computational cluster with 16 CPUs (Intel Xeon Due Core 3.0GHz) running RedHat Linux Enterprise 5 with 48GB main memory. For batch algorithms, due to memory limit, 8,000 training examples are randomly chosen if training data size exceeds 8,000, whereas only 2,000 training examples are used for the epsilon dataset because of its high dimension.

Five-fold cross-validation is executed on training sets to decide the learning rate $\eta_t \in 2^{[-12:10]}$ for online algorithms, the regularized parameter $\lambda \in 2^{[-10:2]}$ for OPAUC and $\lambda \in 2^{[-10:10]}$ for batch algorithms. For OAM_{seq} and OAM_{gra}, the buffer sizes are fixed to be 100 as recommended in (Zhao et al., 2011). For univariate approaches, the weights (i.e., class ratios) are chosen as done in (Kotlowski et al., 2011).

The performances of the compared methods are evaluated by five trials of 5-fold cross validation, where the AUC values are obtained by averaging over these 25 runs. Table 2 shows that OPAUC is significant better than the other four online algorithms OAM_{seq}, OAM_{gra}, online Uni-Exp and online Uni-Squ, particularly for large datasets. The win/tie/loss counts show that OPAUC is clearly superior to these online algorithms, as it wins for most times and never loses. Table 3 shows that OPAUC is highly competitive to the other five batch learning algorithms; this is impressive because these batch algorithms require storing the whole training dataset whereas OPAUC does not store training data. Additionally, batch LS-SVM which optimizes the square loss is comparable to the other batch algorithms, verifying our argument that square loss is effective for AUC optimization.

Table 5: Testing AUC (mean \pm std.) of OPAUCr with online methods on high-dimensional datasets. \bullet/\circ indicates that OPAUCr is significantly better/worse than the corresponding method (pairwise t -tests at 95% significance level). ‘N/A’ means that no result was obtained after running out 10^6 seconds (about 11.6 days).

datasets	sector	sector.lvr	news20	news20.binary	rcv1v2	ecml2012
OPAUCr	.9292 \pm .0081	.9962 \pm .0011	.8871 \pm .0083	.6389 \pm .0136	.9686 \pm .0029	.9828 \pm .0008
OAM _{seq}	.9163 \pm .0087 \bullet	.9965 \pm .0064	.8543 \pm .0099 \bullet	.6314 \pm .0131 \bullet	.9686 \pm .0026	N/A
OAM _{gra}	.9043 \pm .0100 \bullet	.9955 \pm .0059	.8346 \pm .0094 \bullet	.6351 \pm .0135 \bullet	.9604 \pm .0025 \bullet	.9657 \pm .0055 \bullet
online Uni-Exp	.9215 \pm .0034 \bullet	.9969 \pm .0093	.8880 \pm .0047	.6347 \pm .0092 \bullet	.9822 \pm .0042 \circ	.9820 \pm .0016 \bullet
online Uni-Squ	.9203 \pm .0043 \bullet	.9669 \pm .0260	.8878 \pm .0066	.6237 \pm .0104 \bullet	.9818 \pm .0014	.9530 \pm .0041 \bullet
OPAUC ^f	.6228 \pm .0145 \bullet	.6813 \pm .0444 \bullet	.5958 \pm .0118 \bullet	.5068 \pm .0086 \bullet	.6875 \pm .0101 \bullet	.6601 \pm .0036 \bullet
OPAUC ^{CP}	.7286 \pm .0619 \bullet	.9863 \pm .0258 \bullet	.7885 \pm .0079 \bullet	.6212 \pm .0072 \bullet	.9353 \pm .0053 \bullet	.9355 \pm .0047 \bullet
OPAUC ^{pca}	.8853 \pm .0114 \bullet	.9893 \pm .0288 \bullet	.8878 \pm .0115	N/A	.9752 \pm .0020 \circ	N/A

We also compare the running time of OPAUC and the online algorithms OAM_{seq}, OAM_{gra}, online Uni-Exp and online Uni-Squ, and the average CPU time (in seconds) are shown in Figure 1. As expected, online Uni-Squ and online Uni-Exp takes the least time cost because they optimize on single-instance (univariate) loss, whereas the other algorithms work by optimizing pairwise loss. On most datasets, the running time of OPAUC is competitive to OAM_{seq} and OAM_{gra}, except on the mnist and epsilon datasets which have the highest dimension in Table 1.

6.2 Comparison on High-Dimensional Data

Next, we study the performance of using low-rank matrices to approximate the full covariance matrices, denoted by OPAUCr. Six datasets^{4,5} with nearly or more than 50,000 features are used, as summarized in Table 4. The news20.binary dataset contains two classes, different from news20 dataset. The original news20 and sector are multi-class datasets; in our experiments, we randomly group the multiple classes into two meta-classes each containing the same number of classes, and we also use the sector.lvr dataset which regards the largest class as positive whereas the union of other classes as negative. The original ecml2012 and rcv1v2 are multi-label datasets; in our experiments, we only consider the label with the largest population, and remove the features in ecml2012 dataset that take zero values for all instances.

Besides the online algorithms OAM_{seq}, OAM_{gra}, online Uni-Exp and online Uni-Squ, we also evaluate three variants of OPAUC to study the effectiveness of approximating full covariance matrices with low-rank matrices:

- **OPAUC^f**: Randomly selects 1,000-dim features and then works with full covariance matrices;

4. <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/>

5. <http://www.ecmlpkdd2012.net/discovery-challenge>

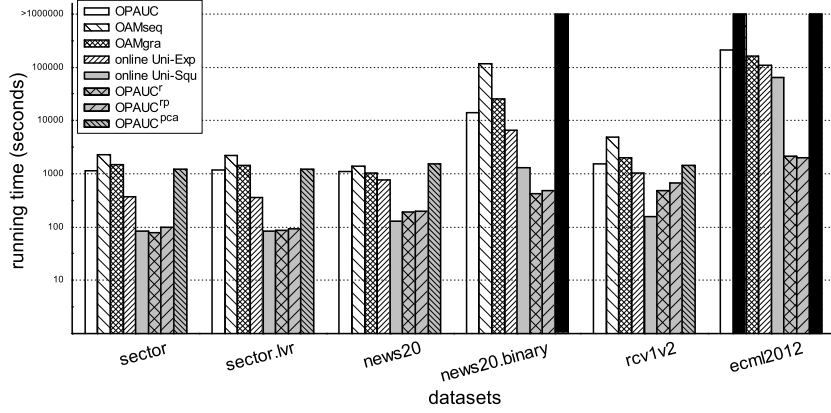
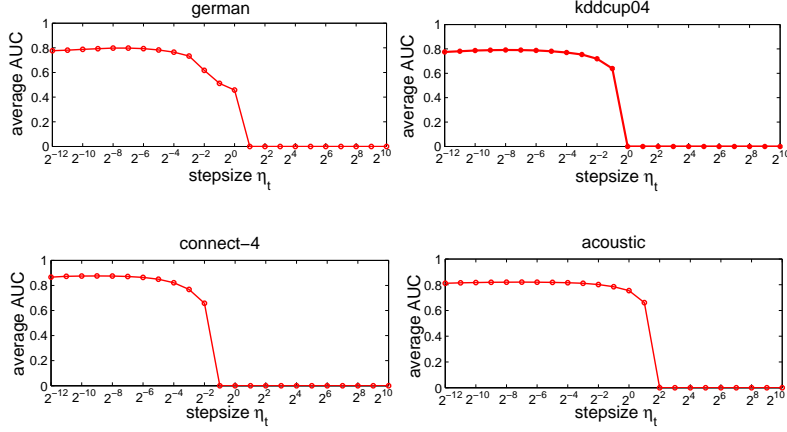
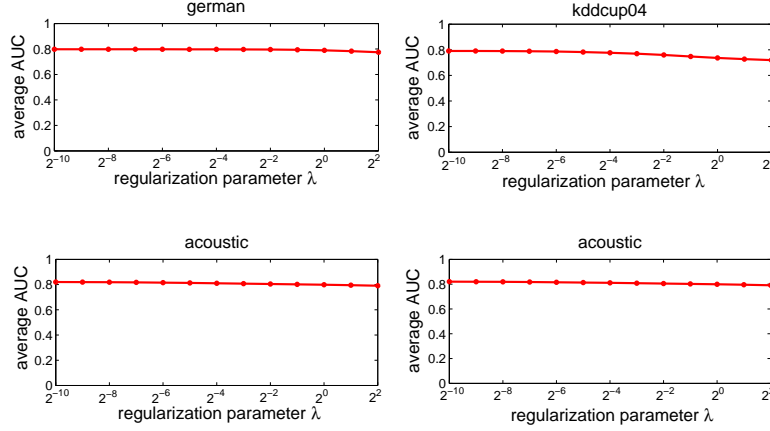


Figure 2: Comparison of the running time on high dimensional datasets. Full black columns imply that no results were returned after running out the maximal running time.

- **OPAUC^{rp}**: Projects into a 1,000-dim feature space by Random Projection, and then works with full covariance matrices;
- **OPAUC^{pca}**: Projects into a 1,000-dim feature space obtained by Principle Component Analysis, and then works with full covariance matrices.

Similar to Section 5.1, five-fold cross validation is executed on training sets to decide the learning rate $\eta_t \in 2^{[-12:10]}$ and the regularization parameter $\lambda \in 2^{[-10:2]}$. Due to memory and computational limit, the buffer sizes are set to 50 for OAM_{seq} and OAM_{gra} , and the rank τ of OPAUCr is also set to 50. The performances of the compared methods are evaluated by five trials of 5-fold cross validation, where the AUC values are obtained by averaging over these 25 runs.

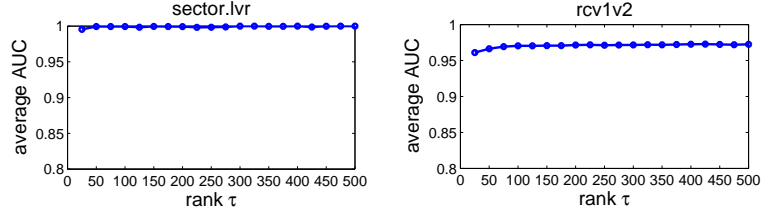
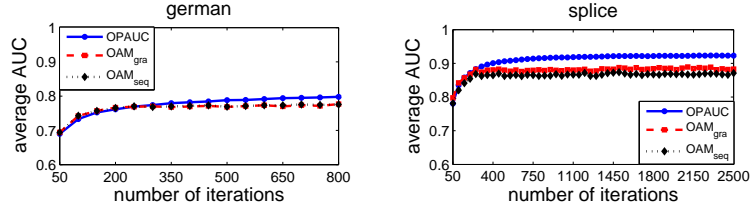
The comparison results are summarized in Table 5 and the average running time is shown in Figure 2. These results clearly show that our approximate OPAUCr approach is superior to the other compared methods. Compared with OAM_{seq} and OAM_{gra} , the running time costs are comparable whereas the performance of OPAUCr is better. Online Uni-Squ and Uni-Exp are more efficient than OPAUCr because it optimizes univariate loss, but the performance of OPAUCr is highly competitive or better, except on rcv1v2 , the only dataset with less than 50,000 features. Compared with the three variants, OPAUC^f and OPAUC^{rp} are more efficient, but with much worse performances. $\text{OPAUC}^{\text{pca}}$ achieves a better performance on rcv1v2 , but it is worse on datasets with more features; particularly, on the two datasets with the largest number of features, $\text{OPAUC}^{\text{pca}}$ cannot return results even after running out 10^6 seconds (almost 11.6 days). Our approximate OPAUCr approach is significantly better than all the other methods (if they return results) on the two datasets

Figure 3: Influence of stepsize η_t Figure 4: Influence of regularization parameter λ

with the largest number of features: `news.binary` with more than 1 million features, and `ecml2012` with nearly 100 thousands features. These observations validate the effectiveness of the low-rank approximation used by OPAUCr for handling high-dimensional data.

6.3 Parameter Influence

We study the influence of parameters in this section. Figure 3 shows that stepsize η_t should not be set to values bigger than 1, whereas there is a relatively big range between $[2^{-12}, 2^{-4}]$ where OPAUC achieves good results. Figures 4 shows that OPAUC is not sensitive to the value of regularization parameter λ given that it is not set with a big value. Figure 5 shows that OPAUCr is not sensitive to the values of rank τ , and it works well even when $\tau = 50$; this verifies Theorem 4 that a relatively small τ value suffices to lead to a good approximation performance. Figure 6 compares studies the influence of the iterations for OPAUC, OAM_{seq} and OAM_{gra} , and it is observable that OPAUC convergence faster than the other two algorithms, which verifies our theoretical argument in Section 4.


 Figure 5: Influence of rank τ

 Figure 6: Convergence comparisons of OPAUC, OAM_{seq} and OAM_{gra}

7. Conclusion

In this paper, we study one-pass AUC optimization that requires going through the training data only once, without storing the entire dataset. Here, a big challenge lies in the fact that AUC is measured by a sum of losses defined over pairs of instances from different classes. We propose the OPAUC approach, which employs a square loss and requires the storing of only the first and second-statistics for the received training examples. A nice property of OPAUC is that its storage requirement is $O(d^2)$, where d is the dimension of data, independent from the number of training examples. To handle high-dimensional data, we develop an approximate strategy by using low-rank matrices. The effectiveness of our proposed approach is verified both theoretically and empirically. In particular, the performance of OPAUC is significantly better than state-of-the-art online AUC optimization approaches, even highly competitive to batch learning approaches; the approximate OPAUC is significantly better than all compared methods on large datasets with one hundred thousands or even more than one million features. An interesting future issue is to develop one-pass AUC optimization approaches not only with a performance comparable to batch approaches, but also with an efficiency comparable to univariate loss optimization approaches.

References

- N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.
- C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems 16*, pages 313–320. MIT Press, Cambridge, MA, 2004.

- P. A. Flach, J. Hernández-Orallo, and C. F. Ramirez. A coherent interpretation of AUC as a measure of aggregated classification performance. In *Proceedings of the 28th International Conference on Machine Learning*, pages 657–664, Bellevue, WA, 2011.
- W. Gao and Z.-H. Zhou. On the consistency of AUC optimization. *CoRR*, 1208.0645v3, 2012.
- Alex Gittens and Joel A. Tropp. Tail bounds for all eigenvalues of a sum of random matrices. *CoRR*, abs/1104.4513v2, 2011.
- J. A. Hanley and B. J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983.
- P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. Society for Industrial and Applied Mathematics, 1987.
- A. Herschtal and B. Raskutti. Optimising area under the ROC curve using gradient descent. In *Proceedings of the 21st International Conference on Machine Learning*, Alberta, Canada, 2004.
- D. Hsu, S. Kakade, and T. Zhang. Random design analysis of ridge regression. In *Proceedings of the 25th Annual Conference on Learning Theory*, pages 9.1–9.24, Edinburgh, Scotland, 2012.
- T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 377–384, 2005.
- T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 217–226, Philadelphia, PA, 2006.
- I. Johnstone. High dimensional statistical inference and random matrices. In *Proceedings of the International Congress of Mathematicians*, pages 307–333, Madrid, Spain, 2006.
- W. Kotlowski, K. Dembczynski, and E. Hüllermeier. Bipartite ranking through minimization of univariate loss. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1113–1120, Bellevue, WA, 2011.
- X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Trans. Systems, Man, and Cybernetics - B*, 39(2):539–550, 2009.
- C. E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 8(4):283–298, 1978.
- Y. Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer, 2003.
- J. Nocedal and S. J. Wright. *Numerical optimization*. Springer-Verlag, 1999.
- F. J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the 15th International Conference on Machine Learning*, pages 445–453, Madison, Wisconsin, 1998.

- A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning*, pages 449–456, Edinburgh, Scotland, 2012.
- C. Rudin and R. E. Schapire. Margin-based ranking and an equivalence between AdaBoost and RankBoost. *Journal of Machine Learning Research*, 10:2193–2232, 2009.
- S. Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, Hebrew University of Jerusalem, 2007.
- N. Srebro, K. Sridharan, and A. Tewari. Smoothness, low noise and fast rates. In *Advances in Neural Information Processing Systems 24*, pages 2199–2207. MIT Press, Cambridge, MA, 2010.
- P. Zhao, S. Hoi, R. Jin, and T. Yang. Online AUC maximization. In *Proceedings of the 28th International Conference on Machine Learning*, pages 233–240, Bellevue, WA, 2011.